A Linguagem de Programação do VisuAlg (3)

Comandos de Repetição

O VisuAlg implementa as três estruturas de repetição usuais nas linguagens de programação: o laço contado `para...ate...faca` (similar ao *for...to...do* do Pascal), e os laços condicionados `enquanto...faca` (similar ao *while...do*) e `repita...ate` (similar ao *repeat...until*). A sintaxe destes comandos é explicada a seguir.

Para ... faça

Esta estrutura repete uma seqüência de comandos um determinado número de vezes.

```
para <variável> de <valor-inicial> ate <valor-limite> [passo
<incremento>] faca
   <seqüência-de-comandos>
fimpara
```

<variável >	É a variável contadora que controla o número de repetições do laço. Na versão atual, deve ser necessariamente uma variável do tipo `inteiro`, como todas as expressões deste comando.
<valor-inicial>	É uma expressão que especifica o valor de inicialização da variável contadora antes da primeira repetição do laço.
<valor-limite >	É uma expressão que especifica o valor máximo que a variável contadora pode alcançar.
<incremento >	É opcional. Quando presente, precedida pela palavra `passo`, é uma expressão que especifica o incremento que será acrescentado à variável contadora em cada repetição do laço. Quando esta opção não é utilizada, o valor padrão de *<incremento>* é 1. Vale a pena ter em conta que também é possível especificar valores negativos para *<incremento>*. Por outro lado, se a avaliação da expressão *<incremento >* resultar em valor nulo, a execução do algoritmo será interrompida, com a impressão de uma mensagem de erro.
fimpara	Indica o fim da seqüência de comandos a serem repetidos. Cada vez que o programa chega neste ponto, é acrescentado à variável contadora o valor de *<incremento >*, e comparado a *<valor-limite >*. Se for menor ou igual (ou maior ou igual, quando *<incremento >* for negativo), a seqüência de comandos será executada mais uma vez; caso contrário, a execução prosseguirá a partir do primeiro comando que esteja após o `fimpara`.

<valor-inicial >, <valor-limite > ***e*** *<incremento >* são avaliados uma **única vez** antes da execução da primeira repetição, e **não se alteram durante a execução do laço**, mesmo que variáveis eventualmente presentes nessas expressões tenham seus valores alterados.

No exemplo a seguir, os números de 1 a 10 são exibidos em ordem crescente.

```
algoritmo "Números de 1 a 10"
var j: inteiro
inicio
para j de 1 ate 10 faca
   escreva (j:3)
fimpara
fimalgoritmo
```

Importante: Se, logo no início da primeira repetição, *<valor-inicial >* for maior que *<valor-limite >* (ou menor, quando *<incremento>* for negativo), o laço não será executado nenhuma vez. O exemplo a seguir não imprime nada.

```
algoritmo "Numeros de 10 a 1 (não funciona)"
var j: inteiro
```

```
inicio
para j de 10 ate 1 faca
   escreva (j:3)
fimpara
fimalgoritmo
```

Este outro exempo, no entanto, funcionará por causa do **passo -1**:

```
algoritmo "Numeros de 10 a 1 (este funciona)"
var j: inteiro
inicio
para j de 10 ate 1 passo -1 faca
   escreva (j:3)
fimpara
fimalgoritmo
```

Enquanto ... faça

Esta estrutura repete uma seqüência de comandos enquanto uma determinada condição (especificada através de uma expressão lógica) for satisfeita.

```
enquanto <expressão-lógica> faca
   <seqüência-de-comandos>
fimenquanto
```

<expressão-lógica>	Esta expressão que é avaliada antes de cada repetição do laço. Quando seu resultado for VERDADEIRO, *<seqüência-de-comandos>* é executada.
fimenquanto	Indica o fim da *<seqüência-de-comandos>* que será repetida. Cada vez que a execução atinge este ponto, volta-se ao início do laço para que *<expressão-lógica>* seja avaliada novamente. Se o resultado desta avaliação for VERDADEIRO, a *<seqüência-de-comandos>* será executada mais uma vez; caso contrário, a execução prosseguirá a partir do primeiro comando após `fimenquanto`.

O mesmo exemplo anterior pode ser resolvido com esta estrutura de repetição:

```
algoritmo "Números de 1 a 10 (com enquanto...faca)"
var j: inteiro
inicio
j <- 1
enquanto j <= 10 faca
   escreva (j:3)
   j <- j + 1
fimenquanto
fimalgoritmo
```

Importante: Como o laço `enquanto...faca` testa sua condição de parada **antes** de executar sua seqüência de comandos, esta seqüência poderá ser executada **zero ou mais vezes**.

Repita ... até

Esta estrutrura repete uma seqüência de comandos até que uma determinada condição (especificada através de uma expressão lógica) seja satisfeita.

```
repita
   <seqüência-de-comandos>
ate <expressão-lógica>
```

repita	Indica o início do laço.

ate *<expressão-lógica>*	Indica o fim da *<seqüência-de-comandos>* a serem repetidos. Cada vez que o programa chega neste ponto, *<expressão-lógica>* é avaliada: se seu resultado for FALSO, os comandos presentes entre esta linha e a linha `repita` são executados; caso contrário, a execução prosseguirá a partir do primeiro comando após esta linha.

Considerando ainda o mesmo exemplo:

```
algoritmo "Números de 1 a 10 (com repita)"
var j: inteiro
inicio
j <- 1
repita
   escreva (j:3)
   j <- j + 1
ate j > 10
fimalgoritmo
```

Importante: Como o laço `repita...ate` testa sua condição de parada **depois** de executar sua seqüência de comandos, esta seqüência poderá ser executada **uma ou mais vezes**.

Comando Interrompa

As três estruturas de repetição acima permitem o uso do comando `interrompa`, que causa uma saída imediata do laço. Embora esta técnica esteja de certa forma em desacordo com os princípios da programação estruturada, o comando `interrompa` foi incluído no VisuAlg por ser encontrado na literatura de introdução à programação e mesmo em linguagens como o Object Pascal (Delphi/Kylix), Clipper, VB, etc. Seu uso é exemplificado a seguir:

```
algoritmo "Números de 1 a 10 (com interrompa)"
var x: inteiro
inicio
x <- 0
repita
   x <- x + 1
   escreva (x:3)
   se x = 10 entao
      interrompa
   fimse
ate falso
fimalgoritmo
```

O VisuAlg permite ainda uma forma alternativa do comando `repita...ate`, com a seguinte sintaxe:

```
algoritmo "Números de 1 a 10 (com interrompa) II"
var x: inteiro
inicio
x <- 0
repita
   x <- x + 1
   escreva (x:3)
   se x = 10 entao
      interrompa
   fimse
fimrepita
fimalgoritmo
```

Com esta sintaxe alternativa, o uso do `interrompa` é obrigatório, pois é a única maneira de se sair do laço `repita...fimrepita`; caso contrário, este laço seria executado indeterminadamente.

Anterior Próxima

Objetivos Tela principal Menu A linguagem do VisuAlg Referências da linguagem do VisuAlg Mais recursos